Computers, Materials & Continua
DOI: 10.32604/cmc.2023.031729
Article

Tech Science Press

Fuzzy Firefly Based Intelligent Algorithm for Load Balancing in Mobile Cloud Computing

Poonam* **and Suman Sangwan**

CSE Department, Deenbandhu Chhotu Ram University of Science and Technology, Murthal, 131039, Sonepat, India
*Corresponding Author: Poonam. Email: ahlawatp12@gmail.com
Received: 25 April 2022; Accepted: 29 June 2022

Abstract: This paper presents a novel fuzzy firefly-based intelligent algorithm for load balancing in mobile cloud computing while reducing makespan. The proposed technique implicitly acts intelligently by using inherent traits of fuzzy and firefly. It automatically adjusts its behavior or converges depending on the information gathered during the search process and objective function. It works for 3-tier architecture, including cloudlet and public cloud. As cloudlets have limited resources, fuzzy logic is used for cloudlet selection using capacity and waiting time as input. Fuzzy provides human-like decisions without using any mathematical model. Firefly is a powerful meta-heuristic optimization technique to balance diversification and solution speed. It balances the load on cloud and cloudlet while minimizing makespan and execution time. However, it may trap in local optimum; levy flight can handle it. Hybridization of fuzzy firefly with levy flight is a novel technique that provides reduced makespan, execution time, and Degree of imbalance while balancing the load. Simulation has been carried out on the Cloud Analyst platform with National Aeronautics and Space Administration (NASA) and Clarknet datasets. Results show that the proposed algorithm outperforms Ant Colony Optimization Queue Decision Maker (ACOQDM), Distributed Scheduling Optimization Algorithm (DSOA), and Utility-based Firefly Algorithm (UFA) when compared in terms of makespan, Degree of imbalance, and Figure of Merit.

Keywords: Cloud computing; cloudlet; mobile cloud computing; fuzzy; firefly; load balancing; makespan; degree of imbalance

1 Introduction

The widespread use of mobile devices has led to the emergence of mobile cloud technology. Mobile users want several applications on their mobiles, and they need cloud services to carry out storage and computation due to limited mobile resources like battery life, storage, and computation. Resource limitations lead to the offloading of mobile data to the cloud to carry out remote computation and then send back the computational result to the mobile device. The mobile cloud computing (MCC) concept introduced in 2010 [1] is a combination of three technologies: Cloud Computing (CC), Mobile

Computing (MC), and Mobile Internet [2,3]. The epitome of MCC is to let the user access and compute accurate and real-time information at any time and any place.

Cloud computing has brought a new era of development to the internet. It has removed all limitations regarding computer applications. Cloud is the interconnected assemblage of high-performance servers with extensive storage and computational power accessible through the internet. Cloud resources are available to users on pay per use basis [4]. The proliferation of wireless mobile communication, new web technologies, and the evolution of mobile internet business are driving forces for the development of the internet. Internet technology is a way of communication by using the internet, which is to provide real-time network resources and services to the users. Mobile internet gives convenience and the facility to provide various new business opportunities, computational services, and quality assurance [5]. MC is about sharing different resources and exchanging data among intelligent devices like mobile phones, laptops, and computers. [6].

Although, MCC has eliminated all limitations of mobile devices related to storage, computation, and battery life. However, connecting mobile devices to the cloud undergoes high network latency and high power consumption while data are offloading specifically through 4G/LTE networks [7]. To overcome these problems concept of the cloudlet framework was introduced by M. Satyanarayanan in 2009 [8]. Cloudlets are trusted, resource-rich computer or a cluster of computers that is well connected to the internet and available for use by nearby mobile devices [8]. Cloudlets provide physical proximity to mobile users. They are deployed just one hop distance compared to the cloud, resulting in the offloading of tasks to the nearest cloudlet [9]. Mobile devices use Wi-Fi to connect to the proximate cloudlet [10]. Cloudlet is the middle layer of the 3-tier architecture, as shown in Fig. 1. Low latency, high bandwidth, and one-hop distance provide mobile devices with real-time interactive services. For large applications requiring high computational and processing capabilities or when cloudlets are already busy, the cloud is used for data offloading and computational purposes [11]. Due to the proliferation of MCC nowadays, it is essential to balance the load on cloudlet and cloud.

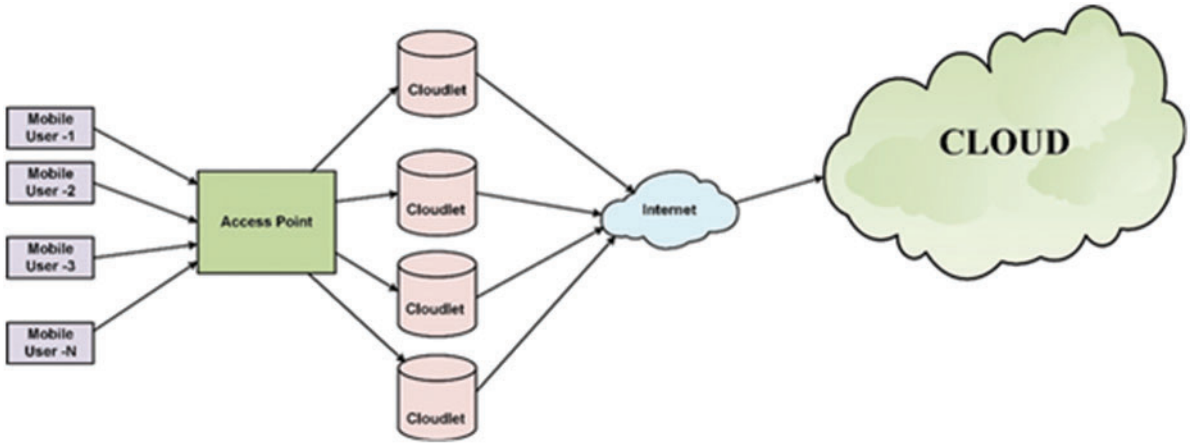


Figure 1: Three-tier architecture

A hybrid meta-heuristic load balancing technique, named fuzzy firefly-based algorithm with levy flight (LF), is introduced in this paper. Firefly algorithm (FA) is a powerful nature-inspired optimization technique used widely by researchers. The essence of the FA is to maintain the proper balance between solution diversification and speed. FA may fall in local optimum and leads to premature convergence. To overcome this issue, LF is used to enhance the exploration of global search

space. Levy distribution provides the random movement by generating a random direction and a step length. Fuzzy logic is used to make cloudlet selection. Fuzzy logic provides evenness among true and false values without using any mathematical model. Optimization using fuzzy logic is simple, fast, and adaptive, leading to system stability.

In recent times, MCC has turned out to be a significant research area as it enhances mobile devices' capabilities by integrating them with cloud computing technologies and cloudlet. The major highlights of this paper are as follows:

- Study of existing load balancing techniques in MCC.
- Propose a novel hybrid intelligent load balancing technique named Fuzzy Firefly Algorithm with Levy Flight for balancing the load on cloud and cloudlet.
- Cloudlet selection using Fuzzy logic.
- Formulation of makespan and Degree of imbalance using a mathematical model.
- Formulation of fitness function using execution model and Degree of imbalance.
- Comparison of the proposed technique with ACOQDM, DSOA, and UFA using makespan and Degree of imbalance.

The remnants of this paper are organized as follows: Section 2 presents existing work on load balancing in MCC. Section 3 drafts the mathematical model of the load balancing problem. The proposed methodology presents a novel hybrid Fuzzy Firefly algorithm with Levy Flight to address the load balancing in Section 5. The results of the proposed technique are discussed in Section 6. Finally, Section7 concludes the paper with some future directions.

2 Related Work

Many researchers have proposed different load balancing and scheduling techniques [12]. Job scheduling in a mobile cloud environment is challenging as there are numerous processors, and all have numerous different or same instances of virtual machines (VM). Assigning jobs to the correct VM and improving the performance and cost is a significant issue in the mobile cloud environment. Job scheduling should also consider allocating the load to all the VMs to achieve load balancing while minimizing the makespan [13]. Many works have been carried out using a few legacy algorithms used for scheduling and load balancing purposes.

In some existing works, scheduling is performed based on the size of tasks like the Min-Min algorithm chose the job with small tasks [14] and came out to be better as it reduced the makespan while balancing the load. An improved min-min scheduling algorithm with load balancing [15] had been proposed, but execution cost was not considered in this paper. It showed an increase in resource utilization and decreasing the makespan. The max-min algorithm chose the maximum size task after selecting the task [16]. Many research works exist on priority-based scheduling [17] to perform fair scheduling to prioritize the jobs. However, these legacy algorithms were not good enough for complex, uncertain, and real-time user demand. None of these works had considered utility-based prioritization.

On the other hand, meta-heuristic and nature-inspired algorithms like Genetic algorithm (GA), Ant colony optimization (ACO), FA, and Bees colony optimization attempt to provide near-optimal solutions for the scheduling and load balancing problem. In [18], ACO is used and compared the makespan with the First Come First Serve (FCFS) and Round Robin (RR) algorithms. Cost and job interdependency were not considered in this work. ACO concentrated on makespan and cost reduction. Its performance depended on the initial solution, and it also had a more extended

convergence than several algorithms. Bees' colony algorithm also had a restriction as it converged only in its local optima. Most of the previous works focused only on the execution time [19].

Guo et al. in [20] used Particle swarm optimization (PSO) to reduce processing time. A multi-objective scheduling algorithm was proposed to reduce the cost and execution time of jobs using the Pareto frontier method. The Pareto frontier method was also used in one more multi-objective PSO algorithm [21] to reduce the execution time of jobs. Here, attention was given to reducing data transfer time. However, load balancing was not considered in any of the abovementioned work. A layered mobile cloud resource scheduling algorithm was proposed for mobile devices in the mobile cloud environment. An energy-efficient and cost-aware mobile service provisioning scheme for the mobile cloud was proposed in [22]. Chunlin et al. [23] studied the composition of mobile device services to enhance user utility within the capacity of physical resources and other restrictions in the mobile grid. The authors of the paper [24] proposed an algorithm for task assignment in a mobile cloud environment while balancing the load. Cloudlets are used for the execution of tasks leading to reduced makespan. However, this algorithm has the highest energy consumption rate. This paper [25] proposed a distributed scheduling algorithm for resource-intensive mobile applications using the Lagrangian method. A bargaining protocol to maximize resource utilization and profit was introduced in [26]. It balanced the load by distributing jobs to reliable VMs using FA. Wang et al. [27,28] provide task-centered resource allocation in mobile edge computing. Gu et al. [29,30] proposed a secure framework for data queries in fog and cloud environments. Liao et al. proposed [31,32,33] task offloading algorithms in edge computing, providing cost savings related to data transmission, latency, and bandwidth usage, among other benefits.

From the above discussion, we observe that less work is done on cloudlet selection and load balancing on the cloudlet and cloud. Cloudlet selection based on remaining capacity is rarely found. Hence, we propose an intelligent hybrid meta-heuristic Fuzzy Firefly Algorithm with Levy Flight for Load Balancing using capacity and waiting time for cloudlet selection to increase resource reliability while reducing the makespan and maintaining load balancing.

3 Mathematical Problem Descriptions

For formulating the problem mathematically, Let us consider n mobile users, each submitting a task set at any time t in the system. This submitted task set is represented by vector $D = \{D_1, D_2, D_3 \ldots\ldots D_n\}$. Tasks are offloaded to cloudlet or public cloud in MCC for execution. Assume p cloudlets in close vicinity of the mobile user represented by vector $C = \{C_1, C_2, C_3 \ldots\ldots\ldots C_p\}$ and m VMs on each cloudlet or cloud represented by vector $VM = \{VM_1, VM_2, VM_3 \ldots\ldots\ldots VM_m\}$. The user request can be offloaded either on cloudlet or public cloud as the user request arrives. Cloudlets have limited resources and computing power as compared to the cloud. If cloudlets are full and refuse to process the tasks, then tasks can be offloaded to the public cloud. We have assumed an assignment matrix A to show the assignment of tasks to VMs on cloudlets. In this matrix, rows represent the tasks, and columns represent the VMs, as shown by Eq. (1). Entry A_{ij} in matrix A will be 1, if task D_i is assigned to VM_j, otherwise it will be 0.

$$A = \begin{bmatrix} A_{11} & A_{12} & A_{13} & \ldots\ldots & A_{1m} \\ A_{21} & A_{22} & A_{23} & \ldots\ldots & A_{2m} \\ \ldots & \ldots & \ldots & \ldots\ldots & \ldots \\ A_{n1} & A_{n2} & A_{n3} & \ldots\ldots & A_{nm} \end{bmatrix} \tag{1}$$

Similarly, a matrix CT represents the completion time of tasks on VMs. Here also, matrix rows represent the tasks, and columns represent the VMs, as shown by Eq. (2).

$$CT = \begin{bmatrix} CT_{11} & CT_{12} & CT_{13} & \ldots\ \ldots & CT_{1m} \\ CT_{21} & CT_{22} & CT_{23} & \ldots\ \ldots & CT_{2m} \\ \ldots & \ldots & \ldots & \ldots\ \ldots & \ldots \\ CT_{n1} & CT_{n2} & CT_{n3} & \ldots\ \ldots & CT_{nm} \end{bmatrix} \tag{2}$$

In this paper, makespan and Degree of imbalance are used for performance analysis of the system. These metrics are defined as follows:

3.1 Completion Time

The total time to execute all the assigned tasks is known as completion time. It is evaluated using Eq. (3).

$$Completion\ Time, T = \sum_{i=1}^{n} \sum_{j=1}^{m} T_{ij}^{c} * A_{ij} \tag{3}$$

where T_{ij}^{c} is the completion time of i^{th} task on j^{th} VM. It is evaluated using Eq. (4)

$$T_{ij}^{c} = T_{ij}^{s} + T_{ij}^{w} + T_{ij}^{tran} \tag{4}$$

where T_{ij}^{s}, T_{ij}^{w} and T_{ij}^{tran} are service time, waiting time, and transmission time of i^{th} task on j^{th} VM respectively.

3.2 Service Time

The total time taken to execute the task is known as service time. It is evaluated using Eq.(5)

$$T_{ij}^{s} = \frac{No.of\ Instructions\ in\ ith\ task}{MIPS\ of\ VMj\ *\ \text{number of core}} \tag{5}$$

3.3 Waiting Time

The time interval between execution start time and arrival time is known as waiting time. It is evaluated using Eq. (6).

$$T_{ij}^{w} = T_{ij}^{st} - T_{ij}^{a} \tag{6}$$

where T_{ij}^{a} represents the arrival time of i^{th} task on j^{th} VM queue and T_{ij}^{st} represents the execution start time of i^{th} task on j^{th} VM.

3.4 Transmission Time

It is the time taken to offload the task to a VM on Cloudlet or public cloud. It is evaluated using Eq. (7).

$$T_{ij}^{tran} = \frac{Bandwidth\ of\ VM_{j}}{Size\ of\ task\ i} \tag{7}$$

where T_{ij}^{tran} is the time taken to offload i^{th} task to j^{th} VM.

3.5 Makespan

Makespan is the maximum completion time of all tasks assigned on VMs. The lowest value of makespan indicates enriched scheduling, and It is evaluated using Eq. (8)

$$Makespan, f_1 = max\left\{ T_{ij}^{c} \middle| \begin{matrix} i \in D, i = 1, 2, \ldots, n \; and \\ j \in VM, j = 1, 2, \ldots m \end{matrix} \right\} \tag{8}$$

3.6 Response Time

It is the time required to give the first response after submitting a request to the system. It is evaluated using Eq. (9). A small value of response time is recommended.

$$Response\ Time,\ T_{ij}^{rs} = T_{ij}^{c} - (T_{ij}^{a} + T_{ij}^{tran}) \tag{9}$$

where T_{ij}^{a} is the arrival time of i^{th} task on j^{th} VM. T_{ij}^{tran} is the transmission time of i^{th} task on j^{th} VM.

3.7 Processing Capacity

It gives the system processing ability equal to the sum of the capacity of all VMs on a cloudlet or public cloud. It is evaluated using Eqs. (10) and (11).

$$\boldsymbol{Processing\ Capacity, PC} = \sum_{\boldsymbol{j=1}}^{\boldsymbol{m}} \boldsymbol{PC_j} \tag{10}$$

$$PC_j = MIPS\ of\ VM_j\ *\ Number\ of\ core\ +\ csbw_j \tag{11}$$

where $csbw_j$ is communication bandwidth handling speed of VM_j.

3.8 Degree of Imbalance (DI)

The Degree of imbalance is measured in terms of the reliability of VMs. The reliability of a VM is its ability to work efficiently under all situations, and it is inversely proportional to load. Reliability of j^{th} VM is evaluated as given in Eq. (12) [34,35].

$$R_j = a_1 \frac{P_{avail}}{P_{req}} + a_2 \frac{M_{avail}}{M_{req}} + a_3 \frac{B_{avail}}{B_{req}} \tag{12}$$

where P_{avail}, M_{avail} and B_{avail} are available computing power, memory, and bandwidth, respectively. P_{req}, M_{req}, B_{req} are required computing power, memory, and bandwidth, respectively. a_1, a_2, a_3 are weights corresponding to power, memory, and bandwidth. Their values are assigned as shown in Eq. (13).

$$\sum_{i-1}^{3} a_i = 1 \tag{13}$$

Degree of imbalance of j^{th} VM is measured as given in Eq. (14).

$$DI_{\mathrm{VMj}} = \left(R_j - R_{avg}\right)^2 \tag{14}$$

where R_{avg} is the average reliability of all VMs and measured in Eq. (15)

$$R_{avg} = \frac{\sum_{j=1}^{m} R_j}{m} \tag{15}$$

Likewise, the Degree of imbalance of all VMs is measured as given in Eq. (16).

$$\textbf{\textit{Degree of imbalance}}, \boldsymbol{f_2} = \sum_{j=1}^{m} \left(\boldsymbol{R_j} - \boldsymbol{R_{avg}}\right)^2 \tag{16}$$

The Degree of imbalance value lies between 0 and 1. It is used to measure the performance or load imbalance of the system, and the scheduler tries to reduce the value of DI.

4 Fitness Function

The ultimate goal of this hybrid load balancing technique is to minimize makespan and Degree of imbalance while balancing the load. So, the fitness function can be represented as follows by Eq. (17) using Eqs. (8) and (16) [35,36]:

$$\textit{Fitness function}, F(x) = w_1 * e^{f_1} + w_2 * e^{f_2} \tag{17}$$

where f_1 and f_2, are objective functions for makespan and Degree of imbalance. Here, e is the exponential function and w_1 and w_2 are weights corresponding to the $f1$ and f_2 s.t. $w_1 + w_2 = 1$ and $w_1, w_2 \in \{0, 1\}$. Weights may be assigned on the basis of the choice or preference of a particular parameter. In the case of equal preference, each weight can be assigned a value of 0.5, so their total should be equal to 1. Hence, the proposed hybrid load balancing technique efficiently balances the mobile cloud load based on the above fitness function.

5 Proposed Methodology

This section presents the concept of Fuzzy, firefly with levy flight, and their hybridization to achieve load balancing on the mobile cloud.

5.1 Fuzzy Based Decision Maker

The Decision-making process is time-consuming, and sometimes regular computations may not find optimal solutions. Fuzzy systems can be successfully applied in areas of decision making and automatic control with good decisions and results within a short time [37]. Lotfi Zadeh developed the fuzzy logic system in 1965 to solve complex problems with incomplete and imprecise data [38]. They can solve problems by deciding like human beings. Fuzzy logic does not have preset limits but has percentile logic. Depending on membership degree, inputs can be assigned true or false, resulting in certainty or uncertainty to the specified set of inputs. Fuzzy logic provides evenness among true and false values without using any mathematical model. Optimization using fuzzy logic is simple, fast, and adaptive, leading to system stability.

The fundamental steps of the fuzzy system are as follows:

- Fuzzification: Converting crisp input values to linguistic terms or fuzzified input
- Fuzzy Inference: Fuzzified output by applying rule base
- Defuzzification: Converting the fuzzy output to crisp values

Let the set of items x are represented by X, where x represents an ordered pair for fuzzy set F shown by Eq. (18).

$$F = \{x, \mu\mathbf{F}(\mathbf{x}) \in \mathbf{X}\} \tag{18}$$

The triangular membership function is used for the fuzzification process, as given by Eq. (19).

$$Triangle\,(x,a,b,c) = \begin{cases} 0, & x < a \\ \dfrac{x-a}{b-a}, & a \le x \ge b \\ \dfrac{c-x}{c-b}, & b \le x \le c \\ 0, & c \le x \end{cases} \tag{19}$$

The proposed method uses a Sugeno fuzzy inference system for cloudlet selection. Inputs are capacity and waiting time, as shown in Fig. 2. Based on the input and rule base shown in Tab. 1, the best cloudlet is selected for processing tasks. Fig. 3 gives a graphical representation of the rule base.

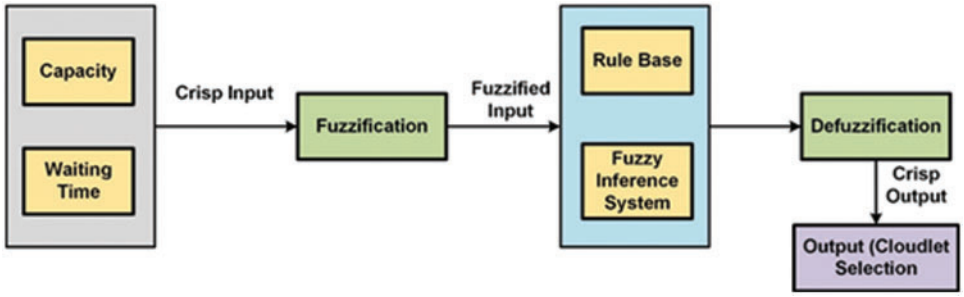


Figure 2: Fuzzy decision maker

Table 1: Fuzzy rule base

Remaining capacity	Waiting time		
	Low	Medium	High
Low	Medium	Low	Very Low
Medium	High	Medium	Very Low
High	Very High	High	Medium

The defuzzification process is about finding a crisp value from the output of the accumulated fuzzy set. If x and y are fuzzy inputs, then fuzzy rule output will be given as by Eq. (20).

$$z = p * x + q * y + r \tag{20}$$

where p, q, and r are constants. The final fuzzy output is the weighted average of all rules output; it is evaluated using Eq. (21).

$$Final\ output = \frac{\sum_1^N w_i * z_i}{\sum_1^N w_i} \tag{21}$$

where N indicates the number of rules in a fuzzy system. w_i represents the weight corresponding to i^{th} Fuzzy rule.

The main objective of FDM is to select a cloudlet for further processing, and the output of FDM is fed as input to the FA for allocating tasks on VM with load balancing.

5.2 Firefly Algorithm

A nature-inspired meta-heuristic technique like FA is a powerful optimization technique for solving various problems. Meta-heuristic algorithms are gaining popularity due to their simplicity, versatility, and high efficiency. One such meta-heuristic technique is firefly which is widely used for problem-solving. A critical factor in the success of this algorithm is a proper balance between solution diversification and speed over another technique [39]. FA was developed by Yaung [40] in late 2007 and 2008. The basic idea of FA is that flies produce flashes, and one firefly is attracted to the brighter one. Attraction depends on brightness. A less bright fly will move towards a brighter one for a couple of flashing flies. If no brighter flies are there than a particular, that fly will move randomly in search space. Some flashing rules of standard FA are as follows:

- All the fireflies are unisex, so one firefly is attracted to another despite the sex.
- Attraction directly depends on brightness, which decreases with distance.
- The objective function determines the brightness of a firefly.

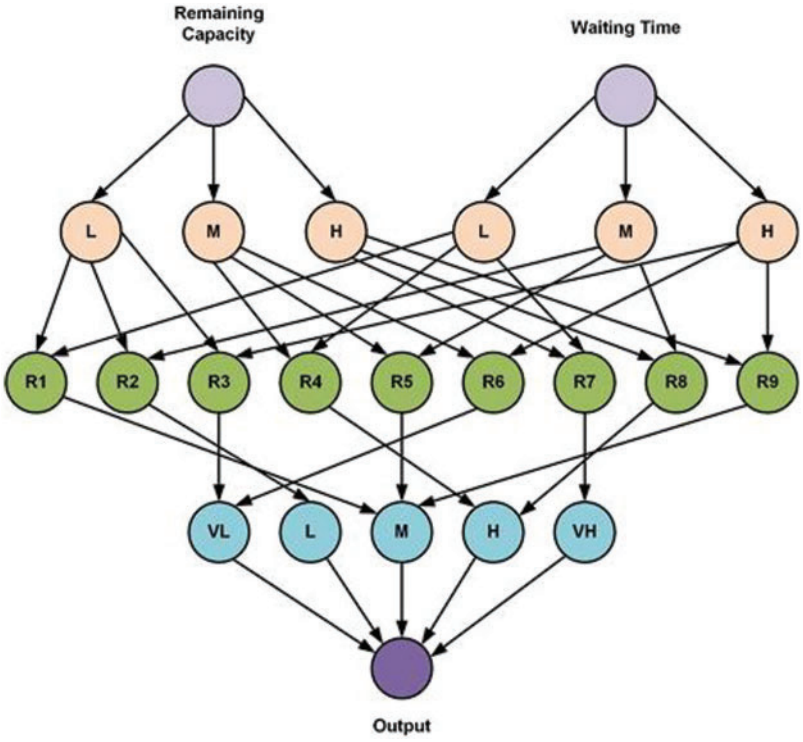


Figure 3: Graphical representation of rule base

5.2.1 Attractiveness and Brightness

Attraction depends on the brightness of the firefly, which is associated with the objective function. A less bright fly will move towards a brighter one for a couple of flashing flies. If no brighter flies are there than a particular, that fly will move randomly in search space. Brightness decreases with distance from the source and is also absorbed by the environment. Brightness or light intensity is given by Eq. (22).

$$I = I_0 e^{-\gamma r^2} \tag{22}$$

where the intensity at the source is I_0 and distance between two flies is r. γ is light absorption coefficient, and it controls the speed of convergence of the solution. Based on it, attractiveness is given by Eq. (23).

$$\beta = \beta_0 e^{-\gamma r^2} \tag{23}$$

where β_0 is attractiveness at source, i.e., $r = 0$.

5.2.2 Distance between Fireflies

Distance between two fireflies i and j at positions x_i and x_j can be defined by Euclidian distance given by Eq. (24) [40].

$$r_{ij} = \sqrt{\sum_{k=1}^{d} (x_{ik} - x_{jk})^2} \tag{24}$$

where x_{ik} is the k^{th} component of spatial coordinate x_j of j^{th} firefly and d represent the number of dimensions. For two dimensions, d is 2, and distance is given by Eq. (25)

$$r_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \tag{25}$$

5.2.3 Positions Update

Position update when firefly x_i moves towards firefly x_j. It is given by Eq. (26).

$$x_{ij} = x_i + \beta e^{-\gamma r^2} \left(X_i - X_j\right) + \alpha \varepsilon_i \tag{26}$$

The first term represents the current position of the firefly, the second term represents attractiveness, and the third term gives random movement. α is the randomization parameter $\alpha \in [0,1]$, and ε gives random distribution, such as uniform distribution.

5.3 Levy Flight

It is observed from the swarm behavior of fireflies [41] that FA often traps into local optimum and converges prematurely. These flies explore the search space through a series of straight flight paths punctuated by a 90° turn, which creates a levy-flight type search pattern. Yaung [40] introduced a levy-flight-based FA to enhance the exploration ability of global search space. It provides randomization through levy distribution instead of conventional uniform distribution. The position update equation is modified to the following Eq. (27).

$$x_i = x_i + \beta e^{-\gamma r^2} \left(X_j - X_i\right) + \alpha \sin(\varepsilon) \otimes Levy \tag{27}$$

The symbol$\otimes$ represents a component-wise multiplication among the random vector from the Levy distribution and the sign vector.

5.4 Fuzzy Firefly Algorithm with Levy Flight (FFALF)

The proposed FFALF provides better results in solving load balancing problems in heterogeneous and platform-independent environments than already existing techniques. This algorithm assigns tasks to reliable VMs. VM's reliability decreases over time as load increases. This algorithm selects a more reliable VM based on fitness function while balancing the load. Fig. 5 represents the proposed algorithm. Firstly, the proposed algorithm checks the cloudlet availability in the vicinity based on waiting time and remaining capacity. A fuzzy decision-maker selects the cloudlet or cloud for task

offloading as this decision-making is fuzzy [42]. A fuzzy system is best for dealing with complex and uncertain problems. The output of FDM is used as an initial population for the FA. Different parameters value used in firefly is shown in Tab. 2 [40,41].

Here, tasks from users arrive arbitrarily, as shown by [43].In our algorithm x_i is solution space represented by an array. Array index designates the task, and array value designates the VM on which the corresponding task has been scheduled. For example, with five tasks and 2 VMs, their mapping is shown in Fig. 4. Tasks 1, 3, and 5 are mapped to the 2nd VM and 2, 5 to the 1st VM.

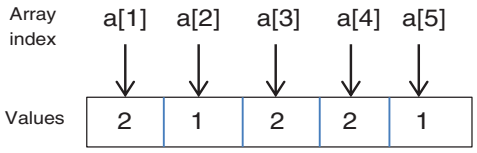


Figure 4: Solution space

Table 2: Parameters values for firefly

Parameter	Value range	Optimal value
Population Size	[50]	50
α	[0.1,10]	0.5
β	[0,1]	0.2
γ	[0,1]	0.5
w_1	[0,1]	0.5
w_2	[0,1]	0.5

After generating the initial population, the fitness of all fireflies is calculated using the objective function shown by Eq. (17). Here, attraction is inversely related to the objective function. A smaller value of the objective function means more attraction [44]. It is observed from the swarm behavior of fireflies [41] that flies follow a logarithmic spiral path in search space. Although this logarithmic path leads to the exploitation of local search space, it traps in the local optimum. Levy flight is used with firefly to find the optimal solution shown by Eq. (27). It helps in coming out of local optimum and provides more search space exploration.

FFALF is a novel algorithm as, firstly, it makes cloudlet selection to provide a faster response to users. In the past, little work is done for cloudlet selection. Secondly, it uses levy flight with the firefly algorithm for uniform task allocation to VMs. Firefly with Levy Flight increases the probability of jumping out of the local optimum and works ideally both in underloading and overloading situations of VMs.

Pseudo-Code: Fuzzy Firefly Algorithm with Levy Flight
Input: Set of VM's
Set of Tasks from Users i.e. initial population
Output: Mapping of Tasks to VM's

1. f(x),X=($X_1, X_2, X_3, \ldots \ldots \ldots X_f$) //Objective function
2. Initialization of γ and $Iter_{max}$ //using Eq.(1) to Eq.(15)
3. F_o=FIS (PC, T^w) // Cloudlet or Cloud selection using Eq.(17), Eq.(18), Eq.(19) and Eq.(20)
4. /* After cloudlet or cloud selection, apply Firefly with Levy Flight to schedule tasks on VM"s with Load balancing */
5. Produce Initial Population of fireflies // given by FDM
6. ***While***($Iter < Iter_{max}$ || terminating condition)
7. ***For*** each VM (firefly) do
8. Compute fitness function $F(x)$ // $F(x)$ corresponds to intensity or attractiveness
9. ***End For***
10. ***For*** each VM_i
11. ***For*** each VM_j
12. ***If*** (I_j>I_i)
13. Move firefly i towards firefly j in all d dimensions
14. $x_i = x_i + \beta e^{-\gamma r^2}\left(X_j - X_i\right) + \alpha \sin(\varepsilon) \otimes Levy$ // using Eq.(26)
15. ***Else***
16. Randomly move firefly i
17. ***End If***
18. Attractiveness changes with distance exponentially // using Eq.(23) and Eq.(24)
19. Find novel solutions and update intensity // using fitness function
20. ***End For***
21. ***End For***
22. Rank the fireflies i.e. VM' and find the current best solution i.e.C^{best} //optimized values
23. $Iter{+}{+}$
24. ***End While***

Figure 5: Fuzzy Firefly Algorithm with Levy Flight (FFALF)

6 Performance Evaluations

In this section, we discuss simulation setup and results obtained in different scenarios using a simulated mobile cloud network.

6.1 Simulation Setup

Simulation has been carried out on a 64-bit Windows 8 machine having Intel Core i3 and 4 GB RAM using the Cloud Analyst tool with Eclipse Java Neon.3 IDE. The proposed work is compared with dynamic algorithms ACOQDM [24], DSOA [25], UFA [26], in two different scenarios, i.e., different number of tasks and different number of VMs. Two datasets, NASA and CLARKNET [24], are used for evaluation purposes. Tab. 3 represents simulation parameters for users, tasks, cloudlet, VMs, and hosts.

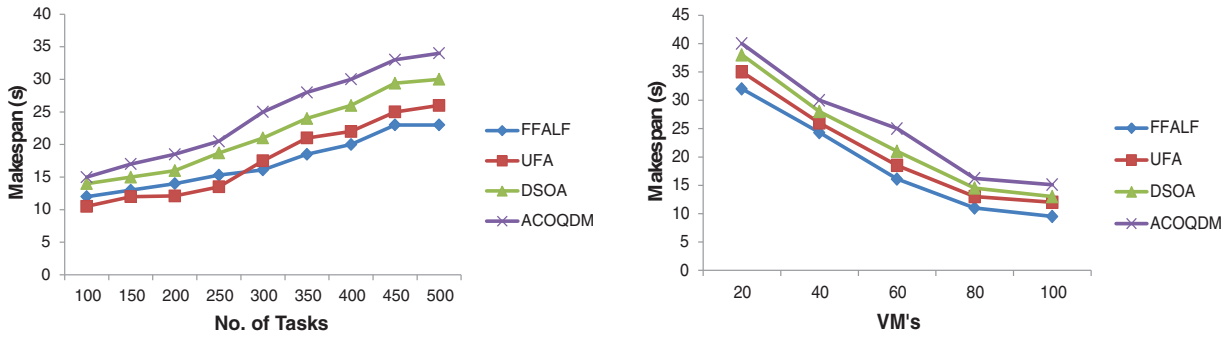
Table 3: Simulation parameters

Entities	Simulation parameters	Range of values
Users	No. of users	10–80
Tasks	No of tasks	100–500
	Size of task	100–500 MI
	Task arrival rate	10–300 requests/sec
Cloudlet	No. of cloudlet	10–15
	No. of servers in cloudlet	1–5
	No. of VMs on cloudlet	5–25
	Bandwidth	100–200 Mbps
	Processing speed	50–300 MIPS
Cloud	No. of VMs on cloud	50–100
	Bandwidth	10000 Mbps
	Processing speed	1000–2000 MIPS
VM	Type of policy	Space shared
	Type of VMM	Xen
	Operating system	Linux
	No of processors	one each

6.2 Results and Discussion

Comparisons are performed in two scenarios: one based on the number of VMs and another based on the number of tasks.

Figs. 6 and 7 show the makespan of the hybrid FFALF algorithm using CLARKNET and NASA datasets, respectively. It is observed that UFA gives a lesser makespan when the number of tasks is smaller. However, FFALF outperforms all when the number of tasks or no. of VMs increases as it allocates tasks to the most reliable VMs.



(a) (b)

Figure 6: Makespan on CLARKNET dataset (a: different no. of tasks with 60 VM, b: different no. of VMs with 300 tasks)

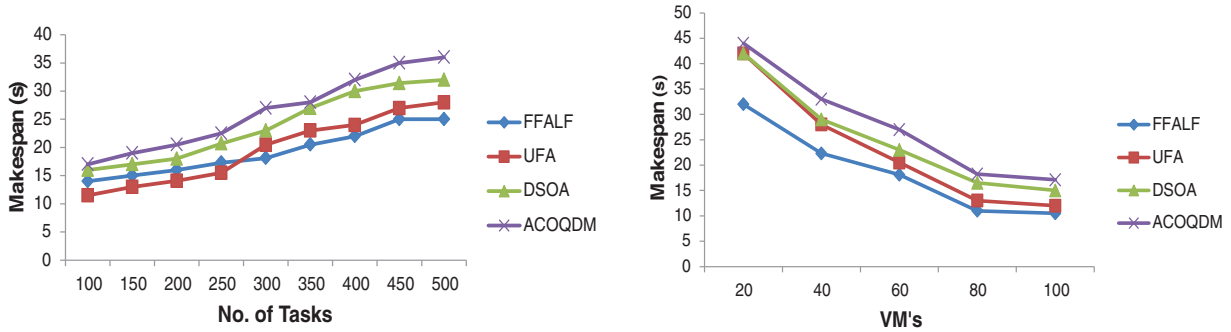


Figure 7: Makespan on NASA dataset (a: different no. of tasks with 60 VM, b: different no. of VMs with 300 tasks)

Fig. 8 shows a degree of imbalance (DI) for the proposed algorithm on two different datasets. A lesser value of DI indicates better load distribution. From Fig. 7, it is observed that the proposed algorithm outperforms all as the tasks increase. In FFALF, cloudlet selection is made based on the remaining resource capacity for task allocation, leading to the uniform load distribution. Further, firefly with levy flight is used for uniform distribution of tasks on cloudlet and cloud.

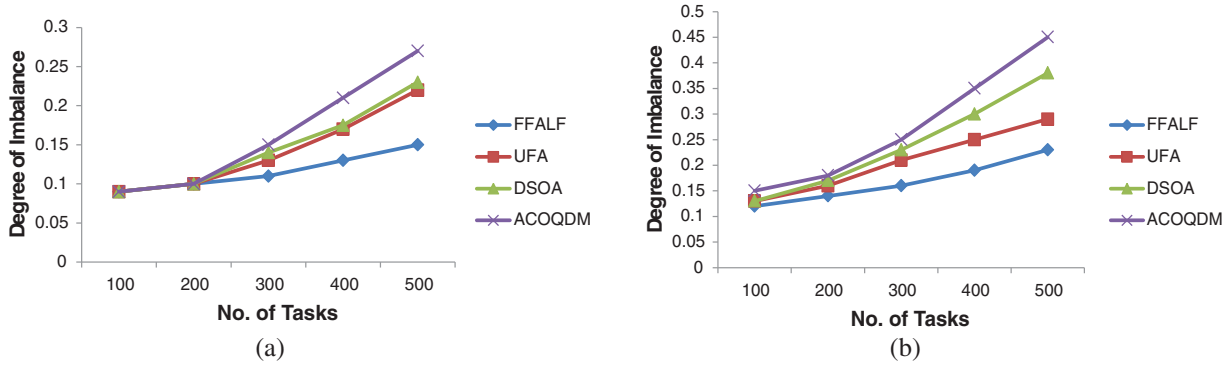


Figure 8: Degree of imbalance (a: on CLARKNET dataset, b: on NASA dataset)

Makespan and Degree of imbalance are considered for performance evaluation of all algorithms. The Figure of Merit is evaluated using Eq. (28), which considers desired parameters in each algorithm. When FFALF is compared with ACOQDM, DSOA, and UFA, It is observed that the FFALF attains higher success rates by strengthening the exploration in the global search space.

$$FoM = \left(\frac{1}{(makespan + DI)}\right) \tag{28}$$

Overall, it can be seen from Fig. 9 that FFALF performs better than the others. In conclusion, FFALF excels UFA, DSOA, and ACOQDM at about 17.39%, 10.20%, and 20%, respectively, on the CLARKNET dataset and 20.42%, 10.98%, 17.91% on the NASA dataset, respectively.

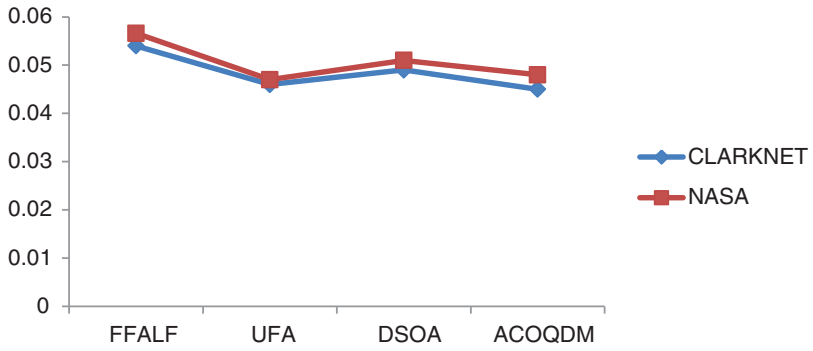


Figure 9: Figure of merit on CLARKNET and NASA dataset

7 Conclusion and Future Work

A novel intelligent hybrid meta-heuristic algorithm for better assignment of offloaded tasks in an MCC environment is proposed in this paper. The proposed technique implicitly acts intelligently as the fuzzy system is an integral part of artificial intelligence, automatically deciding based on input values and rule base. Secondly, Fireflies use swarm intelligence in terms of specific rules derived from their swarm behavior, ensuring interaction between various flies. Thirdly, the proposed algorithm automatically adjusts its behavior or converges depending on the information gathered and provides balance in underloaded and overloaded situations. The simulation was carried out in Cloud Analyst, and performance is compared with UFA, DSOA, and ACOQDM on two different datasets, NASA and CLARKNET. The results show that the FFALF excels in UFA, DSOA, and ACOQDM at about 17.39%, 10.20%, and 20%, respectively, on the CLARKNET dataset and 20.42%, 10.98%, and 17.91% on the NASA dataset while balancing the load. In this algorithm, the mobility of users is not considered, so this algorithm can be best applied in indoor environments like shopping marts, supermarkets, malls, and hospitals, where users are considered stationary. Secondly, this algorithm cannot partition large tasks, which can be distributed among multiple cloudlets simultaneously to achieve parallelism. Hence, in the future, this work can be extended by considering the mobility of users and task partitioning.

Funding Statement: This research work is funded by University Grant Commission with UGC-Ref. No.: 3364/(NET-JUNE 2015).

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] H. T. Dinh, C. Lee, D. Niyato and P. Wang, "A survey of mobile cloud computing: architecture, applications, and approaches," *Wireless Communications and Mobile Computing*, vol. 13, no. 18, pp. 1587–1611, 2013.

[2] L. Zhong, B. Wang and H. Wei, "Cloud computing applied in the mobile internet," in *Proc. ICCSE*, Melbourne, VIC, Australia, pp. 218–221, 2012.

[3] S. S. Qureshi, T. Ahmad, K. Rafique and Shuja-ul-islam, "Mobile cloud computing as future for mobile applications - Implementation methods and challenging issues," in *Proc. ICCCIS*, Beijing, China, pp. 467–471, 2011.

[4] P. Mell and T. Grance, "The NIST definition of cloud computing," *National Institute of Standards and Technology*, vol. 53, no. 6, pp. 50, 2009.
[5] P. Baldoss and G. Thangavel, "Optimal resource allocation and quality of service prediction in cloud," *Computers, Materials & Continua*, vol. 67, no. 1, pp. 253–265, 2021.
[6] E. N. Al-Khanak, S. P. Lee, S. U. R. Khan, N. Behboodian, O. I. Khalaf *et al.,* "A heuristics-based cost model for scientific workflow scheduling in cloud," *Computers, Materials & Continua*, vol. 67, no. 3, pp. 3265–3282, 2021.
[7] Y. Jararweh, L. Tawalbeh, F. Ababneh and F. Dosari, "Resource efficient mobile computing using cloudlet infrastructure," in *Proc. ICMASN*, Dalian, China, pp. 373–377, 2013.
[8] M. Satyanarayanan, P. Bahl, R. Caceres and N. Davies, "The case for vm-based cloudlets in mobile computing," *Pervasive Computing*, vol. 8, no. 4, pp. 14–23, 2009.
[9] K. Dolui and S. K. Datta, "Comparison of edge computing implementations: Fog computing, cloudlet and mobile edge computing," in *Proc. GIoTS*, Geneva, Switzerland, pp. 1–6, 2017.
[10] M. Satyanarayanan, G. Lewis, E. Morris, S. Simanta, J. Boleng *et al.,* "The role of cloudlets in hostile environments," *Pervasive Computing*, vol. 12, no. 4, pp. 40–49, 2013.
[11] C. R. Panigrahi, J. L. Sarkar and B. Pati, "Transmission in mobile cloudlet systems with intermittent connectivity in emergency areas," *Digital Communications and Networks*, vol. 4, no. 1, pp. 69–75, 2018.
[12] A. Arunarani, D. Manjula and V. Sugumaran, "Task scheduling techniques in cloud computing: A literature survey," *Future Generation Computer Systems*, vol. 91, no. 4, pp. 407–415, 2019.
[13] K. Li, "Scheduling parallel tasks with energy and time constraints on multiple many core processors in a cloud computing environment," *Future Generation Computer Systems*, vol. 82, no. 2, pp. 591–605, 2018.
[14] G. Patel, R. Mehta and U. Bhoi, "Enhanced load balanced min-min algorithm for static meta task scheduling in cloud computing," *Procedia Computer Science*, vol. 57, pp. 545–553, 2015.
[15] Poonam and S. Sangwan, "A comparative study of various load balancing algorithms in cloud computing environment," *IJARET*, vol. 11, no. 12, pp. 2735–2760, 2020.
[16] U. Bhoi and P. N. Ramanuj, "Enhanced max-min task scheduling algorithm in cloud computing," *International Journal of Application or Innovation in Engineering and Management*, vol. 2, no. 4, pp. 259–264, 2013.
[17] L. Yang, C. Pan, E. Zhang and H. Liu, "A new class of priority-based weighted fair scheduling algorithm," *Physics Procedia*, vol. 33, pp. 942–948, 2012.
[18] M. Tawfeek, A. El-Sisi, A. Keshk and F. Torkey, "Cloud task scheduling based on ant colony optimization," *The International Arab Journal of Information Technology*, vol. 21, no. 2, pp. 129–137, 2015.
[19] Poonam, Suman and S. Singh, "Load balancing algorithms in cloud computing environment," *International Journal of Advanced Research in Computer Science*, vol. 9, no. 2, pp. 397–401, 2018.
[20] L. Guo, S. Zhao, S. Shen and C. Jiang, "Task scheduling optimization in cloud computing based on heuristic algorithm," *Journal of Networks*, vol. 7, no. 3, pp. 547–553, 2012.
[21] J. Taheri, A. Y. Zomaya, H. J. Siegel and Z. Tari, "Pareto frontier for job execution and data transfer time in hybrid clouds," *Future Generation Computer Systems*, vol. 37, pp. 321–334, 2014.
[22] L. Chunlin, Y. Xin and L. LaYuan, "Flexible service provisioning based on context constraint for enhancing user experience in service oriented mobile cloud," *Journal of Network and Computer Applications*, vol. 66, no. 1, pp. 250–261, 2016.
[23] L. Chunlin and L. LaYuan, "Cost and energy aware service provisioning for mobile client in cloud computing environment," *The Journal of Supercomputing*, vol. 71, no. 4, pp. 1196–1223, 2015.
[24] S. Rashidi and S. Sharifian, "A hybrid heuristic queue based algorithm for task assignment in mobile cloud," *Future Generation Computer Systems*, vol. 68, no. 2, pp. 331–345, 2017.
[25] L. Chunlin, T. Jianhang and L. Youlong, "Distributed QoS-aware scheduling optimization for resource-intensive mobile application in hybrid cloud," *Cluster Computing*, vol. 21, no. 2, pp. 1331–1348, 2018.
[26] M. T. Tapale, R. H. Goudar, M. N. Birje and R. S. Patil, "Utility based load balancing using firefly algorithm in cloud," *Journal of Data, Information and Management*, vol. 2, no. 4, pp. 215–224, 2020.

[27] J. Wang, W. Wu, Z. Liao, Y. W. Jung and J. U. Kim, "An enhanced PROMOT algorithm with D2D and robust for mobile edge computing," *Journal of Internet Technology*, vol. 21, no. 5, pp. 1437–1445, 2020.
[28] Q. Tang, K. Wang, Y. Song, F. Li and J. H. Park, "Waiting time minimized charging and discharging strategy based on mobile edge computing supported by software-defined network," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 6088–6101, 2019.
[29] K. Gu, N. Wu, B. Yin and W. Jia, "Secure data query framework for cloud and fog computing," *IEEE Transactions on Network and Service Management*, vol. 17, no. 1, pp. 332–345, 2019.
[30] W. J. Li, Z. Y. Chen, X. Y. Gao, W. Liu and J. Wang, "Multimodel framework for indoor localization under mobile edge computing environment," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4844–4853, 2018.
[31] Z. Liao, J. Peng, B. Xiong and J. Huang, "Adaptive offloading in mobile-edge computing for ultra-dense cellular networks based on genetic algorithm," *Journal of Cloud Computing*, vol. 10, no. 1, pp. 1–16, 2021.
[32] Z. Liao, J. Peng, J. Huang, J. Wang, J. Wang *et al.,* "Distributed probabilistic offloading in edge computing for genabled massive internet of things," *IEEE Internet of Things Journal*, vol. 8, no. 7, pp. 5298–5308, 2020.
[33] Z. Liao, Y. Ma, J. Huang, J. Wang and J. Wang, "HOTSPOT: A UAV-assisted dynamicmobility-aware offloading for mobile-edge computing in 3-D space," *IEEE Internet of Things Journal*, vol. 8, no. 13, pp. 10940–10952, 2021.
[34] M. Shojafar, S. Javanmardi, S. Abolfazli and N. Cordeschi, "FUGE: A joint meta-heuristic approach to cloud job scheduling algorithm using fuzzy theory and a genetic method," *Cluster Computing*, vol. 18, no. 2, pp. 829–844, 2015.
[35] M. Haris and S. Zubair, "Mantaray modified multi-objective Harris hawk optimization algorithm expedites optimal load balancing in cloud computing," *Journal of King Saud University-Computer and Information Sciences*, vol. 12, no. 10, pp. 3256, 2021.
[36] H. Wu and K. Wolter, "Stochastic analysis of delayed mobile offloading in heterogeneous networks," *IEEE Transactions on Mobile Computing*, vol. 17, no. 2, pp. 461–474, 2017.
[37] N. Mansouri, B. M. H. Zade and M. M. Javidi, "Hybrid task scheduling strategy for cloud computing by modified particle swarm optimization and fuzzy theory," *Computers & Industrial Engineering*, vol. 130, no. 5, pp. 597–633, 2019.
[38] L. A. Zadeh, "Fuzzy sets," *Information and Control*, vol. 8, no. 3, pp. 338–353, 1965.
[39] X. S. Yang, "Nature-inspired metaheuristic algorithms: success and new challenges," *J Comput. Eng. Inf. Technol*, vol. 1, no. 1, pp. 1–3, 2012.
[40] X. S. Yang, Introduction. In: *Nature-inspired metaheuristic algorithms*, 2nd ed., Frome, UK: Luniver Press, pp. 1–9, 2010.
[41] J. Wu, Y. G. Wang, K. Burrage, Y. C. Tian, B. Lawson *et al.,* "An improved firefly algorithm for global continuous optimization problems," *Expert Systems with Applications*, vol. 149, no. 4, pp. 113340, 2020.
[42] A. Islam, A. Kumar, K. Mohiuddin, S. Yasmin, M. A. Khaleel *et al.,* "Efficient resourceful mobile cloud architecture (mRARSA) for resource demanding applications," *Journal of Cloud Computing*, vol. 9, no. 1, pp. 1–21, 2020.
[43] M. Kumar and Suman, "Hybrid cuckoo search algorithm for scheduling in cloud computing," *CMC-Computers, Materials & Continua*, vol. 71, no. 1, pp. 1641–1660, 2022.
[44] B. Patel and B. Patle, "Analysis of firefly-fuzzy hybrid algorithm for navigation of quad-rotor unmanned aerial vehicle," *Inventions*, vol. 5, no. 3, pp. 48, 2020.